INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.680

Corrigendum 3

(02/2001)

SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

OSI networking and system aspects – Abstract Syntax Notation One (ASN.1)

Corrigendum 3:

Summary

This technical corrigendum to ITU-T Rec. X.680 | ISO/IEC 8824-1:

a) introduces new easier value notation for REAL;

b) allows ASN.1 to express what the type within a BIT STRING or OCTET string type is;

c) allows ASN.1 to express what the encoding of the value within an OCTET STRING or BIT STRING type is;

d) allows ASN.1 to specify a pattern for the characters that can appear in a restricted character string type;

e) introduces the Encoding Control Notation;

f) corrects an example so that it uses valid object identifier values;

g) modifies F3.2.10bis (Semantic Model) for new value notation for REAL.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – ABSTRACT SYNTAX NOTATION ONE (ASN.1): SPECIFICATION OF BASIC NOTATION

DRAFT TECHNICAL CORRIGENDUM 3 TO X.680

1) Introduction

In the Introduction (page vii), replace the second and third paragraphs (beginning with "Although this standard notation" and "Outside the context of OSI", respectively) with the following text :

In some protocol architectures, each message is specified as the binary value of a sequence of octets. However, standards-writers need to define quite complex data types to carry their messages, without concern for their binary representation. In order to specify these data types, they require a notation that does not necessarily determine the representation of each value. ASN.1 is such a notation. This notation is supplemented by the specification of one or more algorithms called **encoding rules** that determine the value of the octets that carry the application semantics (called the **transfer syntax**). ITU-T Rec.X.690 | ISO/IEC 8825-1 and ITU-T Rec. X.691 | ISO/IEC 8825-2 specify two families of standardized encoding rules, called **Basic Encoding Rules (BER)** and **Packed Encoding Rules (PER)**. Some users wish to redefine their legacy protocols using ASN.1, but cannot use standardized encoding rules because they need to retain their existing binary representations. Other users wish to have more complete control over the exact layout of the bits on the wire (the transfer syntax). These requirements are addressed by ITU-T Rec.X.692 | ISO/IEC 8825-3 (expected to be approved late 2001) which specify an **Encoding Control Notation (ECN)** for ASN.1. ECN enables designers to formally specify the abstract syntax of a protocol using ASN.1, but to then (if they so wish) take complete or partial control of the bits on the wire by writing an accompanying ECN specification (which may reference standardized Encoding Rules for some parts of the encoding).

There is increasing recognition of the notion of an abstract value of some class (type) divorced from the details of any particular encoding. In order to correctly interpret the bit-pattern representation of the value, it is necessary to know (usually from the context), the class (type) of the value being represented, as well as the encoding mechanism being employed. Thus, the identification of a type is an important part of this Recommendation | International Standard.

2) Subclause 2.1

Add the following item at the end of clause 2.1:

– ITU-T Recommendation X.692 (2001) | ISO/IEC 8825-3:2001 (expected to be approved late 2001), *Information technology – ASN.1 encoding rules: Encoding Control Notation for(ECN) for ASN.1*.

3) Subclause 3.8

Add a new clause 3.8.15 bis as follows:

3.8.15bis contents constraint: A constraint on a bit string or octet string type that specifies either that the contents are to be an encoding of a specified ASN.1 type, or that specified procedures are to be used to produce the contents.

4) Clause 4

In clause 4 add the following line after the line that defines DNIC:

ECN Encoding Control Notation of ASN.1

5) New subclause 11.8bis

Add a new subclause 11.8bis as follows:

11.8bis Real number item

Name of item - realnumber

A "realnumber" shall consist of an integer part that is a series of one or more digits, and optionally a decimal point (.). The decimal point can optionally be followed by a fractional part which is one or more digits. The integer part, decimal point or fractional part (which ever is last present) can optionally be followed by an e or E and an optionally signed exponent which is one or more digits. The leading digit of the integer part shall not be zero unless immediately followed by the decimal point. The leading digit of the exponent shall not be zero unless the exponent is a single digit.

NOTE 1 – This means that it is not possible to represent the value zero under these rules. The representation for zero is specified in clause 20.6 as the single digit "0".

NOTE 2 – The "realnumber" item is always mapped to an real value by interpreting it as decimal notation.

6) Subclause 11.18

Add the reserved words "CONTAINING" and "ENCODED" to both the list of reserved words and to Note 2, and "PATTERN" to the list of reserved words.

7) Subclause 20.5 Note 1

Change Note 1 to read:

NOTE 1 – Non-zero values represented by "base" 2 and by "base" 10 are considered to be distinct abstract values even if they evaluate to the same real number value, and may carry different application semantics.

8) Subclause 20.6

Change subclause 20.6 to read:

20.6 The value of a real type shall be defined by the notation "RealValue":

```
RealValue ::=  NumericRealValue | SpecialRealValue

NumericRealValue ::=  0              |
            realnumber               |
            "-" realnumber |
            SequenceValue  -- Value of the associated sequence type

SpecialRealValue ::=
            PLUS-INFINITY | MINUS-INFINITY
```

The form "0" shall be used for zero values. The alternate form for "NumericRealValue" shall not be used for zero values.

9) New subclause 20.7

Add a new subclause 20.7:

20.7 When the "realnumber" notation is used it identifies the corresponding "base" 10 abstract value. If the RealType is constrained to "base" 2, the "realnumber" identifies the "base" 2 abstract value corresponding either to the decimal value specified by the "realnumber" or to a locally-defined precision if an exact representation is not possible.

10) Subclause 38.1

Add "UTF8String, " after "UniversalString,".

11) Subclause 38.4

Add " and UTF8String" after "UniversalString".

12) Subclause 48.1

Add the following to the end of SubtypeElements:

PatternConstraint

and at the end of the line "InnerTypeConstraints", add "|".

Add a new column to Table 6 with the heading "PatternConstraint", with an entry of "Yes a)" for "Restricted Character String Types" and "No" for all other rows.

Add "UTF8String, " after "VisibleString" to footnote a) of Table 6.

13) Subclause 48.9

Add a new clause 48.9 as follows:

48.9 Pattern constraint

48.9.1 The "PatternConstraint" notation shall be:

PatternConstraint ::= PATTERN Value

48.9.2 "Value" shall be a "cstring" of type UniversalString (or a reference to such a character string) which contains a regular expression as defined in Annex I. The "PatternConstraint" selects those values of the parent type that satisfy the regular expression. The entire value shall satisfy the entire regular expression, i.e. the "PatternConstraint" does not select values whose leading characters match the (entire) regular expression but which contain further trailing characters.

NOTE – "Value" is formally defined as a value of type UniversalString, but the sets of values of type UniversalString and UTF8String are the same (see 36.13). Thus a totally equivalent definition could have been to say that "Value" is a value of type UTF8String.

14) Annex C, subclause C.2.8

Change the two C.2.8 example object identifiers to use values assigned under the "asn1(1) examples(123)" arc:

```
packedBCDStringAbstractSyntaxId OBJECT IDENTIFIER ::=
        { joint-iso-itu-t asn1(1) examples(123) packedBCD(2) charSet(0) }

packedBCDStringTransferSyntaxId OBJECT IDENTIFIER ::=
        { joint-iso-itu-t asn1(1) examples(123) packedBCD(2) characterTransferSyntax(1) }
```

15) Annex F, subclause F.3.2.2 to Amendment 2 (Semantic model)

Add the following after item c:

c bis) Any occurrence of "realnumber" shall be transformed to a "base" 10 associated "SequenceValue". Any occurrence of the "RealValue" associated with "SequenceValue" shall be transformed to the associated "SequenceValue" of the same "base", such that the last digit of the mantissa is not zero.

16) Annex G

Add the following to the end of SubtypeElements:

PatternConstraint

and at the end of the line "InnerTypeConstraints" add a "|".

Also, add the following to Annex G after the line which begins with "PresenceConstraint ::=".

PatternConstraint ::= PATTERN Value

17) New Annex I

Add the following as a new Annex I.

Annex I

ASN.1 regular expressions

(This annex forms an integral part of this Recommendation | International Standard)

Note: This annex will be assigned a new letter, so that it precedes all informative annexes when this Recommendation | International Standard is re-published.

I.1 Definition

I.1.1 A regular expression is a pattern that describes a set of strings whose format conforms to this pattern. A regular expression is itself a string; it is constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions. The smallest expressions, which are (usually) made of one or two characters, are placeholders that stand for a set of characters.
The regular expressions presented here are very similar to those of scripting languages like Perl and to those of XML Schemas, where some other examples of use can be found.

I.1.2 Most characters, including all letters and digits, are regular expressions that match themselves.

EXAMPLE
The regular expression "fred" matches only the string "fred".

I.1.3 Two regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating two substrings that respectively match the concatenated subexpressions.

I.2 Metacharacters

I.2.1 A metacharacter sequence (or metacharacter) is a set of one or more contiguous characters that have a special meaning in the context of a regular expression. The following list contains all of the metacharacter sequences. Their meaning is explained in the following clauses.

[ ]		Match any character in the set where ranges are denoted by "-". A "^" after the opening bracket complements the set which follows it.
{g,p,r,c}		Quadruple which identifies a character of ISO 10646-1 (see 36.7)
\N{name}		Match the named character (or any character of the named character set) as defined in 37.1
.		Match any character (unless it is one of the newline characters defined in 11.1.6)
\d		Match any digit (equivalent to "[0-9]")
\w		Match any alphanumeric character (equivalent to "[a-zA-Z0-9]")
\t		Match the HORIZONTAL TABULATION (9) character (see 11.1.6)
\n		Match any one of the newline characters defined in 11.1.6
\r		Match the CARRIAGE RETURN (13) character (see 11.1.6)
\s		Match any one of the white-space characters defined in 11.1.6
\b		Match a word boundary
\	(prefix)	Quote the next metacharacter and cause it to be interpreted literally
\\		Match the BACKSLACH (92) character
""		Match the double quote character (")
|	(infix)	Alternative between two expressions
( )		Grouping of the enclosed expression
*	(postfix)	Match the previous expression zero, one or several times
+	(postfix)	Match the previous expression one or several times
?	(postfix)	Match the previous expression once or not at all
#n	(postfix)	Match the previous expression exactly n times (where n is a single digit)
#(n)	(postfix)	Match the previous expression exactly n times
#(n,)	(postfix)	Match the previous expression at least n times
#(n,m)	(postfix)	Match the previous expression at least n but not more than m times
#(,m)	(postfix)	Match the previous expression not more than m times

NOTE 1 – The characters CIRCUMFLEX ACCENT (94) "^" and HYPHEN-MINUS (45) "-" are additional metacharacters in certain positions of the string defined in I.2.2.
NOTE 2 – The value in round brackets after a character name in this Annex is the decimal value of the character in ISO 10646.
NOTE 3 – This notation does not provide the metacharacters "^" and "$" to match the beginning and the end of a string respectively. Hence a string shall match a regular expression in its entirety except if the latter includes ".*" at its beginning, at its end or at both sides.
NOTE 4 – The following metacharacter sequences cannot contain a white-space (see 3.8.75) except if the white-space appears immediately prior to or following an end of line:

{g,p,r,c}
\N{name}
#n
#(n)
#(n,)
#(n,m)
#(,m)

In case a regular expression spans more that one line of text, white-space that appears immediately prior to or following the end of line has no significance and matches nothing (see 11.11.1).

I.2.2 A list of characters enclosed by "[" and "]" matches any single character in that list. If the first character of the list is the caret "^", then it matches any character which is not in the list. A range of characters may be specified by giving the first and last characters, separated by a hyphen (according to the order relation defined in 38.4). All metacharacter sequences, except "]" and "\", loose their special meaning inside a list. To include a literal CIRCUMFLEX ACCENT (94) "^", place it anywhere except in the first position or precede it with a backslash. To include a literal HYPHEN-MINUS (45) "-", place it first or last in the list, or precede it with a backslash. To include a literal CLOSING SQUARE BRACKET (93) "]", place it first. If the first character in the list is the caret "^", then the characters "-" and "]" also match themselves when they immediately follow that caret. The metacharacter sequences defined in I.2.3, I.2.4, I.2.6 and I.2.7 can be used between the square brackets where they keep their meaning.

EXAMPLES
The regular expression "[0123456789]", or equivalently "[0-9]", matches any single digit.
The regular expression "[^0]" matches any single character except 0.
The regular expression "[\d^.-]" matches any single digit, a caret, a hyphen or a period.

I.2.3 To avoid any ambiguity between two ISO 10646-1 characters which have the same glyph, two notations are provided. A notation of the form "{group,plane,row,cell}" references a (single) character according to the "Quadruple" production defined in clauses 36.9 and 36.10.

I.2.4 A notation of the form "\N{valuereference}" matches the referenced character if "valuereference" is a reference to a restricted character string value of size 1 (see 36) which is defined or imported in the current module. A notation of the form "\N{typereference}" matches any character of the referenced character set if "typereference" is a reference to a subtype of a "RestrictedCharacterStringType" which is defined in the current module, or is one of the "RestrictedCharacterStringType"s defined in 36.
NOTE – In particular, "valuereference" or "typereference" can be one of the references defined in the module ASN1-CHARACTER-MODULE (see 37.1) and imported into the current module (see 36.7).

EXAMPLES
The regular expression "\N{greekCapitalLetterSigma}" matches GREEK CAPITAL LETTER SIGMA.
The regular expression "\N{BasicLatin}" matches any (single) character of the BASIC LATIN character set.
"[\N{BasicLatin}\N{Cyrillic}\N{BasicGreek}]+", or equivalently "(\N{BasicLatin}|\N{Cyrillic}|\N{BasicGreek})+", are regular expressions that match a string made of any (non null) number of characters from the three character sets specified.

I.2.5 The period "." matches any single character, unless it is one of the newline characters defined in 11.1.6.

I.2.6 The symbol "\d" is a synonym for "[0-9]", i.e. it matches any single digit. The symbol "\t" matches the HORIZONTAL TABULATION (9) character. The symbol "\w" is a synonym for "[a-zA-Z0-9]", i.e. it matches any single (lower-case or upper-case) character or any single digit.

EXAMPLE
The regular expression "\w+(\s\w+)*\." matches a sentence made of at least one (alphanumeric) word. The words are separated by one whitespace character. There is no whitespace character before the ending period.

I.2.7 The symbol "\r" matches the CARRIAGE RETURN (13) character. The symbol "\n" matches any one of the newline characters defined in 11.1.6. The symbol "\s" matches any one of the white-space characters defined in 11.1.6. The symbol "\b" matches the empty string at the beginning or at the end of a word.

NOTE – The characters defined to represent white-space in 11.1.6 include all characters defined to represent newline in 11.1.6.

EXAMPLE

The regular expression ".*\bfred\b.*" matches any string which includes the **word** "fred" (this word is not only a series of fourth characters; it is delimited). Hence it matches strings like "fred" or "I am fred the first", but not strings like "My name is freddy" or "I am afred I don’t know how to spell ‘afraid’!".

I.2.8 A character that normally functions as a metacharacter can be interpreted literally by prefixing it with a "\". If the regular expression includes a double quote (") ASCII QUOTATION MARK (34), this character shall be represented by a pair of double quotes.

EXAMPLES

The regular expression "\." matches the (single) string ".", but not any string of any single character.

The regular expression """" matches the string which contains a single double quote.

The regular expression "\)" matches the string ")".

The regular expression "\a" matches the character "a".

Note – The fourth example shows that the backslash is allowed to precede characters that are not metacharacters, but this use is deprecated (because other metacharacters could be allowed in future versions of Recommendation | International Standard).

I.2.9 Two or more regular expressions may be joined by the infix operator "|". The resulting regular expression matches any string matching either subexpression.

I.2.10 A regular expression may be followed by a repetition operator. If the operator is "?", the preceding item is optional and matched at most once. If the operator is "*", the preceding item will be matched zero or more times. If the operator is "+", the preceding item will be matched one or more times. If the operator is of the form "#(n)", the preceding item is matched exactly n times; in this particular case, the parentheses can be omitted if n consists of one digit. If it is of the form "#(n,)", the item is matched n or more times. If it is of the form "#(,m)", the item is optional and is matched at most m times. Finally, if it is of the form "#(n,m)", the item is matched at least n times, but not more than m times.

NOTE – It is illegal to use the metacharacters "*", "+", "?" or "#" as the first character of a regular expression. It is also illegal to use the metacharacters "#" or "|" as the last character of a regular expression.

EXAMPLES

A phone number like "555-1212" is matched by the regular expression "\d#3-\d#4", or equivalently "\d#(3)-\d#(4)".

A price in dollars like "$12345.90" is matched by the regular expression "$\d#(1,)(\.\d#(1,2))?". Note that parentheses are requested after the "#" symbol when it is followed by a range.

A social security number like "123-45-5678" is matched by the regular expression "\d#3-?\d#2-?\d#4".

I.2.11 Repetition (see I.2.10) takes precedence over concatenation (see I.1.3), which in turn takes precedence over alternation (see I.2.9). A whole subexpression may be enclosed in parentheses to override these precedence rules.

I.2.12 When a regular expression contains subexpressions in parentheses, each (non-quoted) opening parenthesis is successively assigned a distinct (strictly positive) integer from the left to the right of the regular expression**.** Each subexpression can then be referenced inside a comment with a notation like "\1", "\2" which uses the associated integer. The empty subexpression "()" is not permitted.

EXAMPLE

"((\d#2)(\d#2)(\d#4))" -- \1 is a date in which \2 is the month, \3 the day and \4 the year.

NOTE – There is a requirement for formal reference to subexpressions of a regular expression for many purposes. One such instance is the need to write text to document the regular expression within the ASN.1 module. This is a notation which can be used to provide such references. This notation is not used elsewhere in this Recommendation | International Standard.